

# Comparison between SecVtx and TStnSVF

Shawn Kwang  
Mel Shochet

University of Chicago

2008-10-08

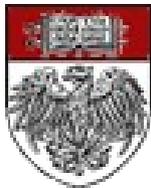


Shawn Kwang

# Introduction

- ▶ The Motivation for this algorithm
- ▶ SecVtx in a nutshell
- ▶ Comparison between Stntuple Secondary Vertex Finder (TStnSVF) algorithm and SecVtx
  - ▶ Tagging match or mismatch
  - ▶ Histograms showing the differences

2008-10-08

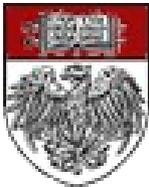


Shawn Kwang

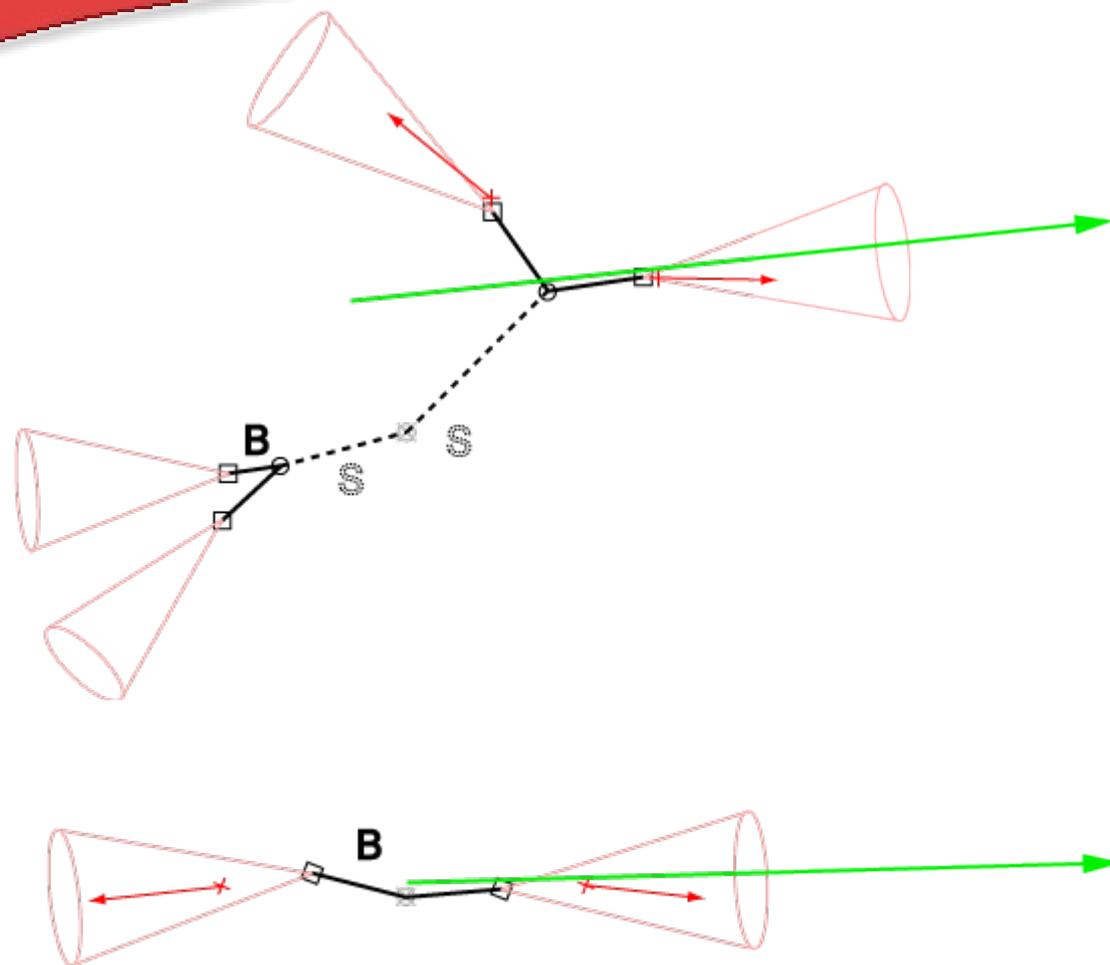
# Motivation

- ▶ Currently the only way to run SecVtx (that I know of) uses CDF Production as input.
  - ▶ Most of us use Ntuples and not the raw Production as the primary method of performing analyses.
    - ▶ It takes too long to run over a large number of production data on tape.
  - ▶ Most of the time this is sufficient
    - ▶ Most analyses use high-pt b-tagging as a tool or variable to select events.
    - ▶ The top group for example uses SecVtx to tag jets as coming from a b-quark or not.
- ▶ My analysis in brief:
  - ▶ We are looking for events with a long-lived ( $c\tau \sim 1\text{cm}$ ) objects decaying into two quarks.
  - ▶  $H_0 \rightarrow a_0, a_0 \rightarrow b\bar{b}$ ,  $b\bar{b}$  is the model we are using for our signal.
    - ▶ Higgs is SM higgs.
    - ▶ The  $a_0$  represents a heavy pseudo-scalar with a long lifetime but is not directly detected.
- ▶ However SecVtx was not designed to look for highly displaced vertexes.
  - ▶ SecVtx as a maximum  $d_0$  cut on tracks it selects for vertexing:  $d_0 < 0.15\text{ cm}$ .
  - ▶ This has the effect of eliminating most of our signal. (Diagram on next page)

2008-10-08



# Motivation



- ▶ Top diagram is the signal, bottom diagram is a typical dijet  $b\bar{b}$  event.
- ▶ Primary vertex is the gray X in a circle.
  - ▶ S is the “ $a_0$ ” (the heavy pseudo-scalar).
- ▶ A  $d_0$  cut on a track (in green) would remove tracks from any signal.

2008-10-08



# Motivation, Cont.

- ▶ It is very time consuming to re-ntuple Stntuple.
- ▶ Thus reprocessing CDF Production data is not feasible My estimate was that it would take nearly 2 years to reprocess the datasets that I need for my analysis.
  - ▶ At minimum we need the ZBB trigger path, the single-tower 5/10 trigger path, and a number of relevant QCD bb, QCD dijet, etc. Monte Carlo.
- ▶ Analyzing Stntuples would be much faster, allow for quicker “turnaround,” and consume less disk space.
  - ▶ Ntuple processing is much faster because there are fewer algorithms being run.
  - ▶ Any bugs in the code would **not** result in another multi-year reprocessing effort.
  - ▶ There is no intermediate step where custom Stntuples are created and need to be stored somewhere.
- ▶ Stntuple has a front end to CTVMFT (Tctvmft.hh & .cc) which allow users to vertex tracks.
  - ▶ It is just a question of writing an algorithm to reproduce what SecVtx does.

2008-10-08

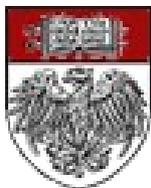


Shawn Kwang

# SecVtx overview

- ▶ The SecVtx algorithm in a nutshell:
- ▶ SecVtx finds a primary vertex in the event.
  - ▶ It uses the best ZVertex vertex as the “seed” vertex.
  - ▶ It uses the PrimVtx Finder algorithm to find a vertex using said seed vertex, constraining the result to the beamline.
- ▶ Next it selects tracks for vertexing.
  - ▶ There are numerous track cuts: z0, d0, z0 Significance, d0 Significance, silicon hits, etc.
  - ▶ Tracks are flagged as pass1 or pass2, the latter having more stringent requirements, but is a subset of the former.
- ▶ Vertexing is performed using two strategies: pass1 and pass2
  - ▶ Seed vertexes are formed using each pair of pass1 tracks.
  - ▶ Tracks are added to the vertex based on the d0 Significance of the additional track and the seed vertex location. At least three tracks in total must be used before a vertex is declared.
  - ▶ If no pass1 vertex is found, pass2 tracks are vertexed together.
  - ▶ Two tracks minimum are required for pass2 vertexing
- ▶ Regardless the vertex is then pruned of tracks that contribute a chi-squared deemed too high, the quantity depending on whether the algorithm is using the loose, tight, or ultratight cuts.
- ▶ The vertex decay length (L2d Significance) is cut on, among other variables.
- ▶ Vertices are checked not to be from a Kshort or Lambda, and not in the material of the detector.

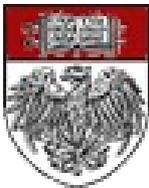
2008-10-08



# Differences Between Algorithms

- ▶ Stntuple does not contain 100% of the information that Production has, and thus my algorithm and SecVtx will have differences.
  - ▶ Si databases are not available (or at least not readily available) and thus some track quality cuts cannot be reproduced.
  - ▶ However, these wind up being second-order effects.
- ▶ In addition, my algorithm does not recalculate the primary vertex location.
  - ▶ Instead it mimics SecVtx by asking the ZVertex algorithm for the best class 12 vertex, and finding the closest PrimeVertex Finder vertex from this “seed” vertex.
  - ▶ Functionally, this has the result of asking for the best vertex class 12 in the PrimeVertex Finder block since the two almost always find the same vertex.

2008-10-08



# Comparison

- ▶ I ran over one fileset of ezbbbj data, 299,167 events. This ZBB data has a large number of b-quarks and thus has good statistics for my test.
  - ▶ Both tagging algorithms were run with the same tight-level cuts.
- ▶ Matched tags are jets where both taggers find a +/-1 tag. Missing is where my algorithm misses a tag that SecVtx finds. Extra is the opposite.

Status	Number of Jets	Percentage(%) of Total
Missing Tags	8324	5.06
Matched Tags	149174	90.7
Extra Tags	6964	4.23

- ▶ Missing tags can be investigated further by looking at events where there are multiple primary vertices.
  - ▶ Events with multiple vertices, where I get the wrong vertex, will have few or no pass1/2 tracks since the  $z_0/d_0$  will be calculated from the wrong vertex.

Status	Number of Jets	Percentage(%) of Total
multiple class12 primary vertices, 0 pass1/2 tracks	81	0.97
multiple class12 primary vertices, 1 pass1/2 tracks	213	2.56
less than 2 pass1/2 tracks, 1 class12 primary vertex	114	1.37
all other jets	7916	95.1

- ▶ The above table shows that the missing tags are not from events where TStnSVF does not find the correct primary vertex.
- ▶ However, the overall match rate is very good (90%).

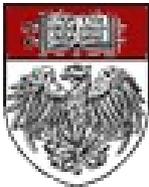
2008-10-08



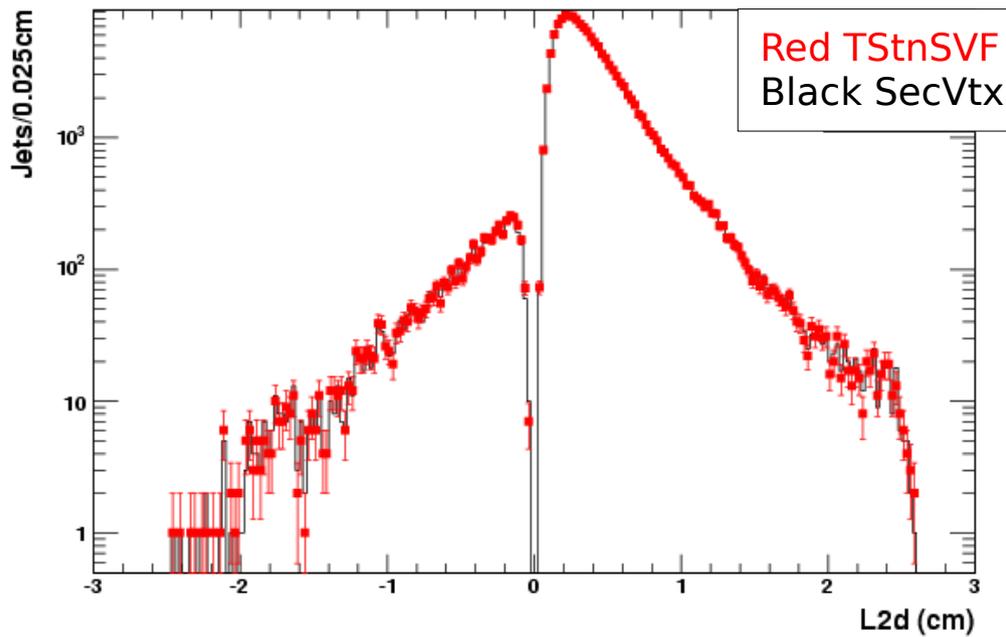
# Comparison

- ▶ For jets were the algorithms matched, I plot some variables to show agreement.
  - ▶ L2d, L2d Significance, Vertex Mass,
- ▶ I also plotted  $1 - (TStnSVF)/(SecVtx)$  to look at the differences between the two algorithms.
  - ▶ Red - TStnSVF
  - ▶ Black - SecVtx
  - ▶ "CDV" was the working name for the algorithm in development.

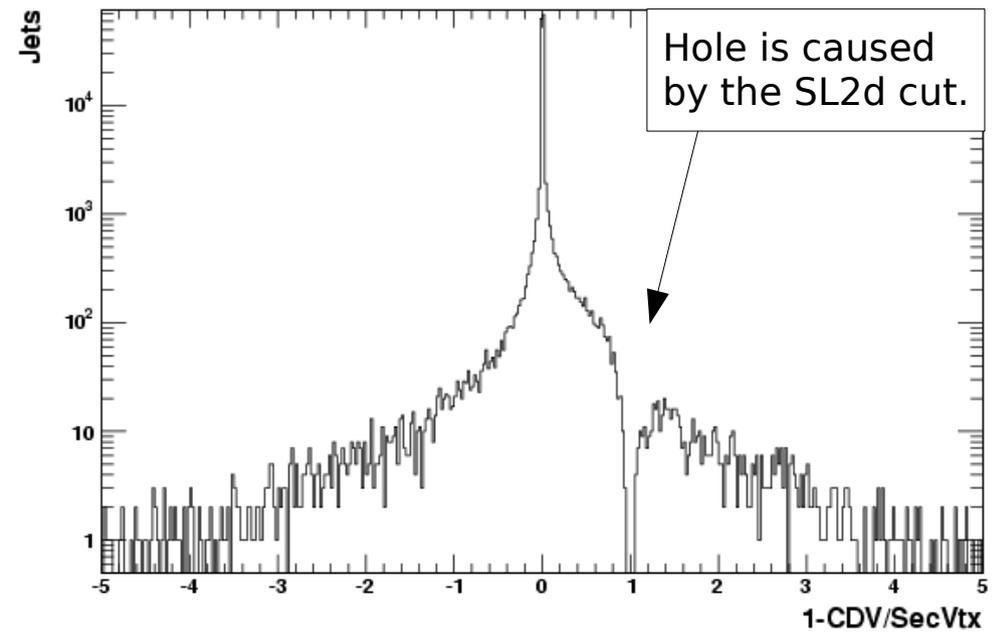
2008-10-08



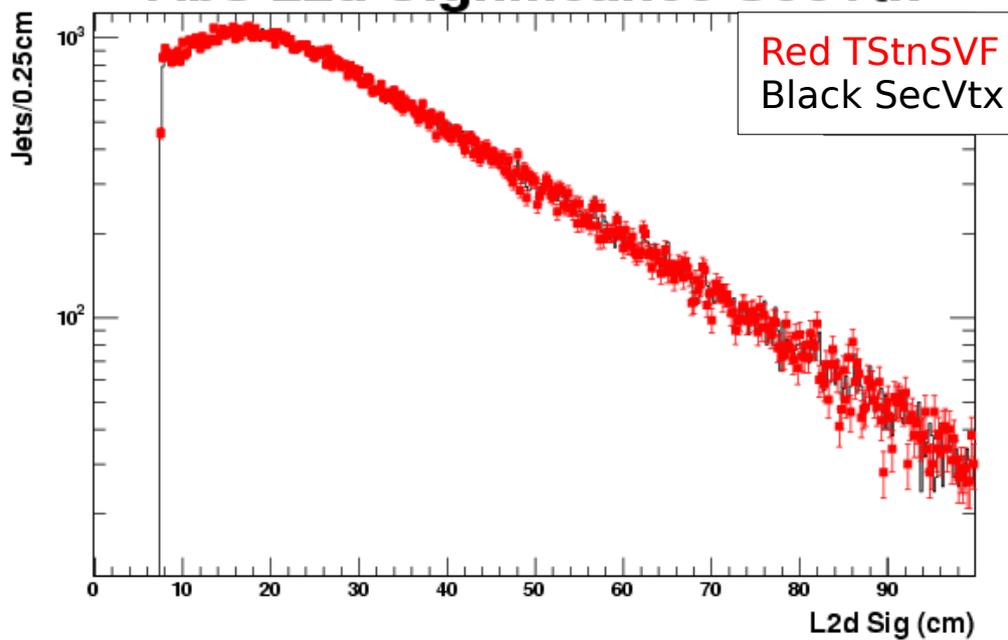
## L2d SecVtx



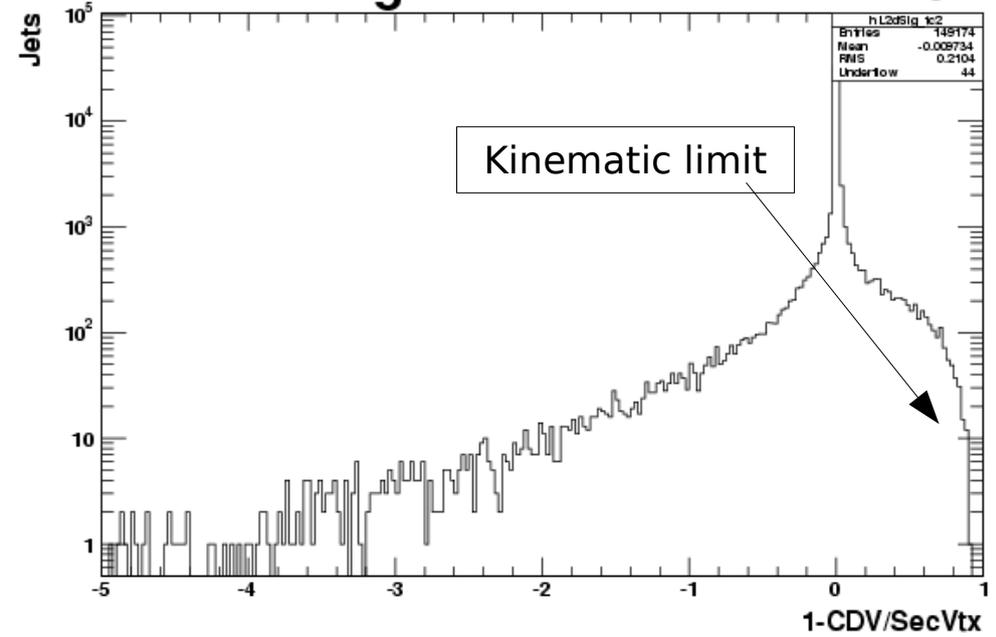
## L2d 1-CDV/SecVtx



## Abs L2d Significance SecVtx



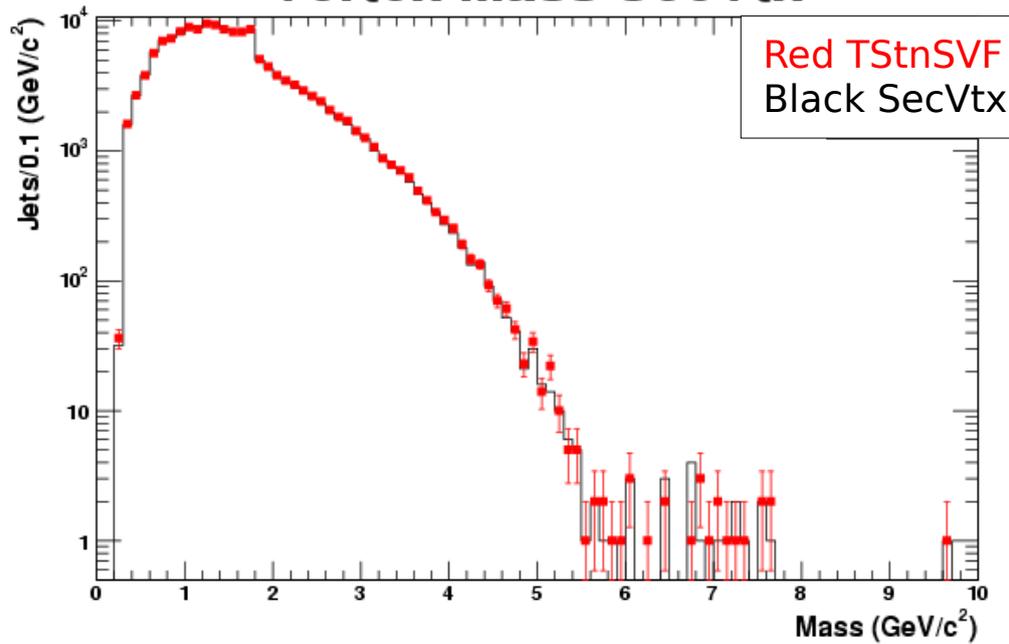
## Abs L2d Significance 1-CDV/SecVtx



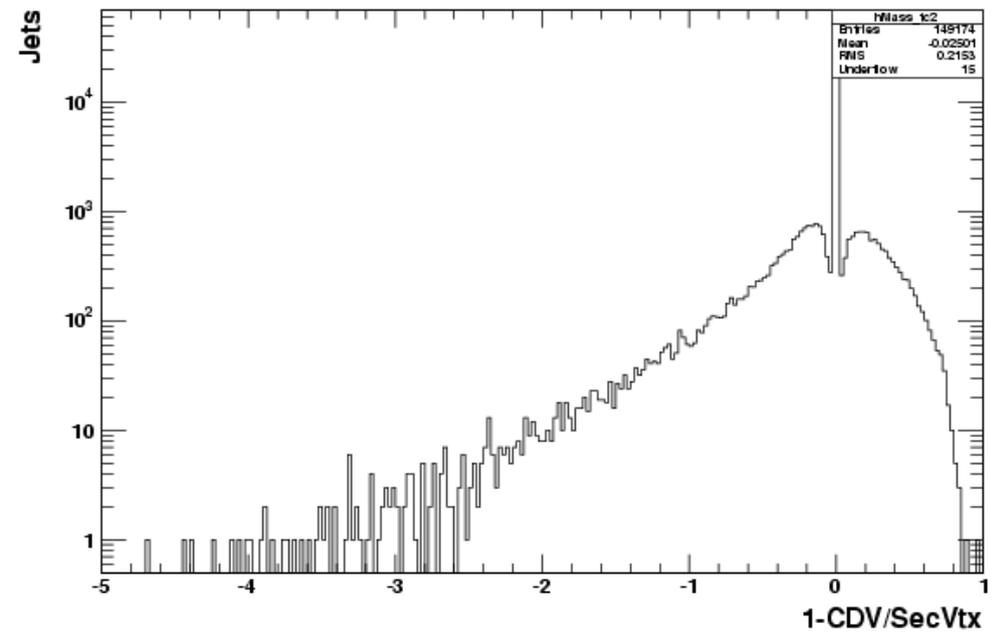
CDV was the working name for the algorithm.



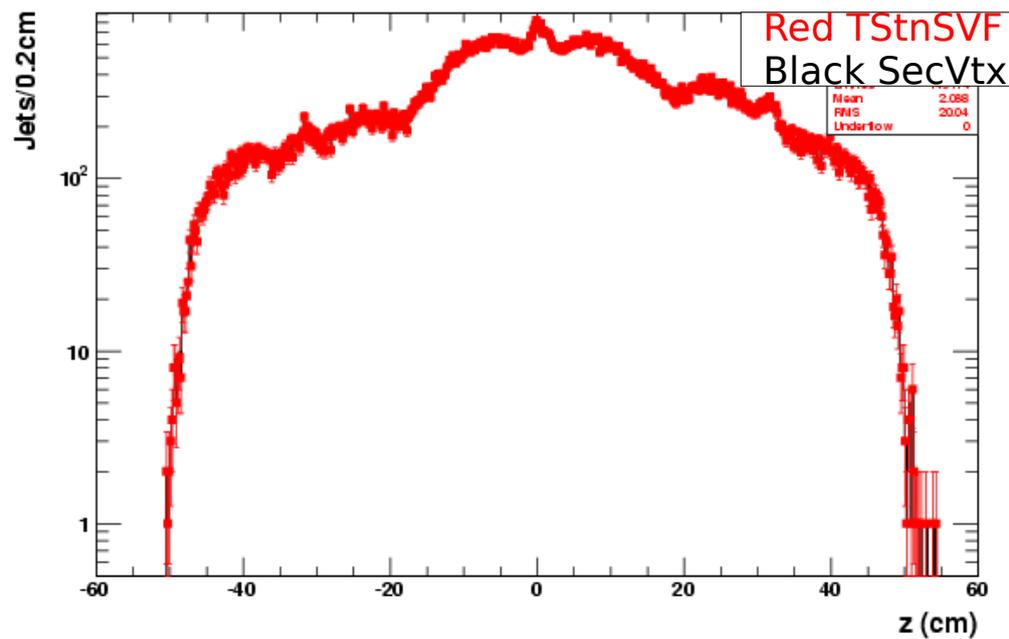
## Vertex Mass SecVtx



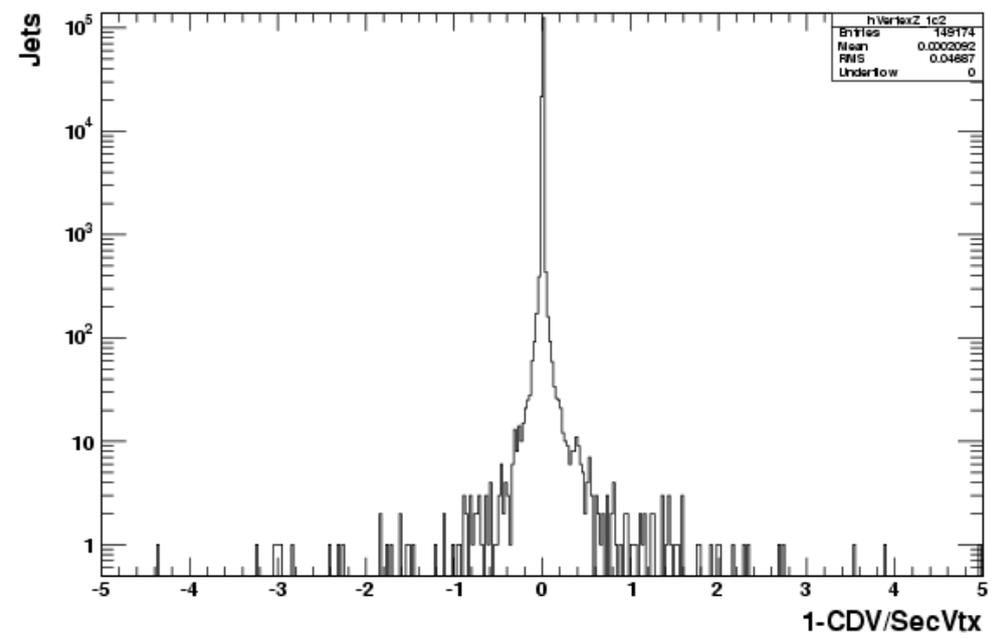
## Vertex Mass 1-CDV/SecVtx



## Vertex Z SecVtx



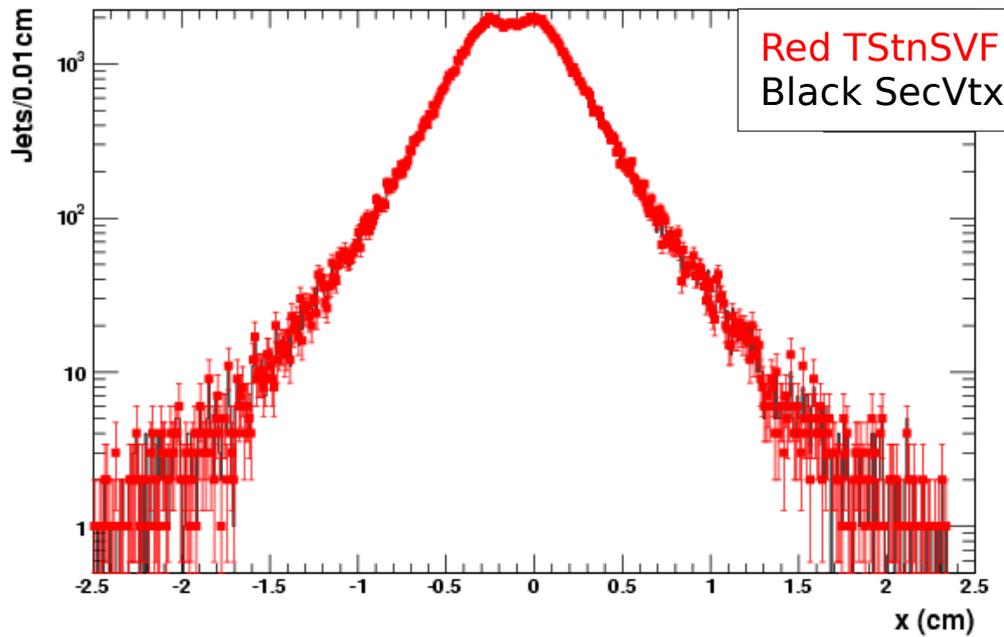
## Vertex Z 1-CDV/SecVtx



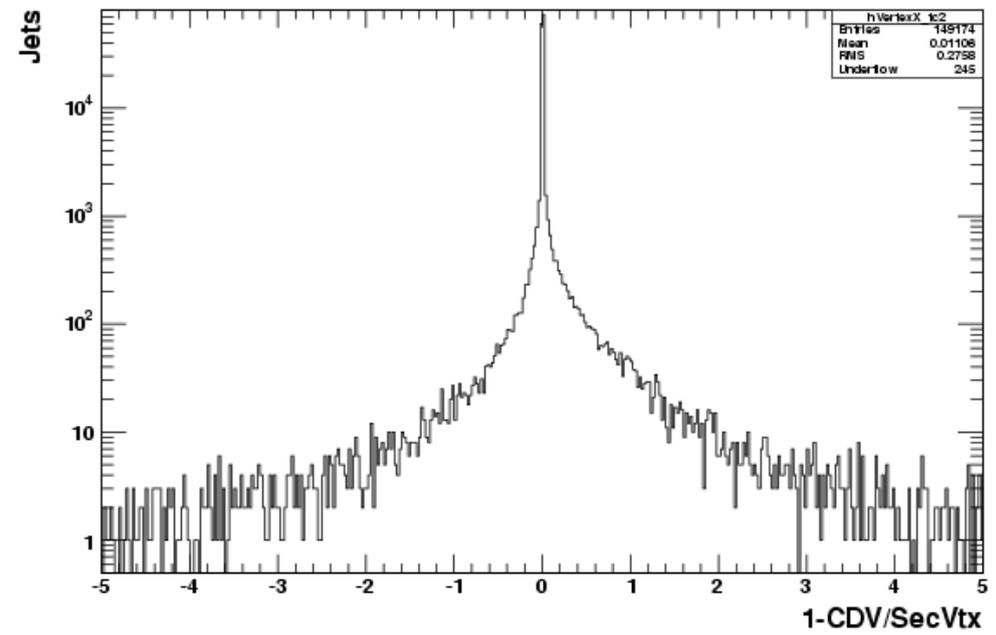
CDV was the working name for the algorithm.



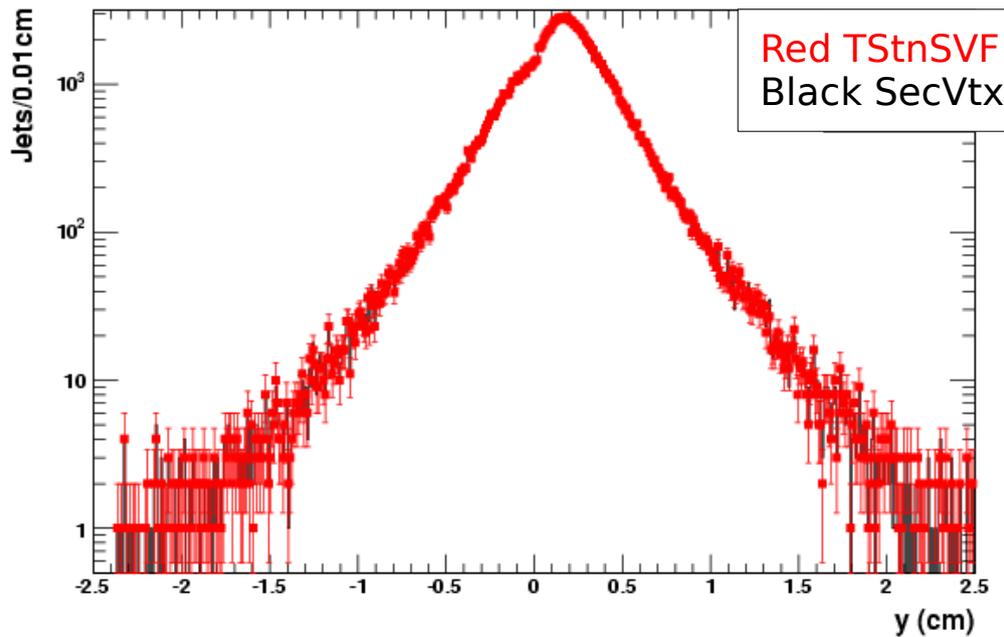
## Vertex X SecVtx



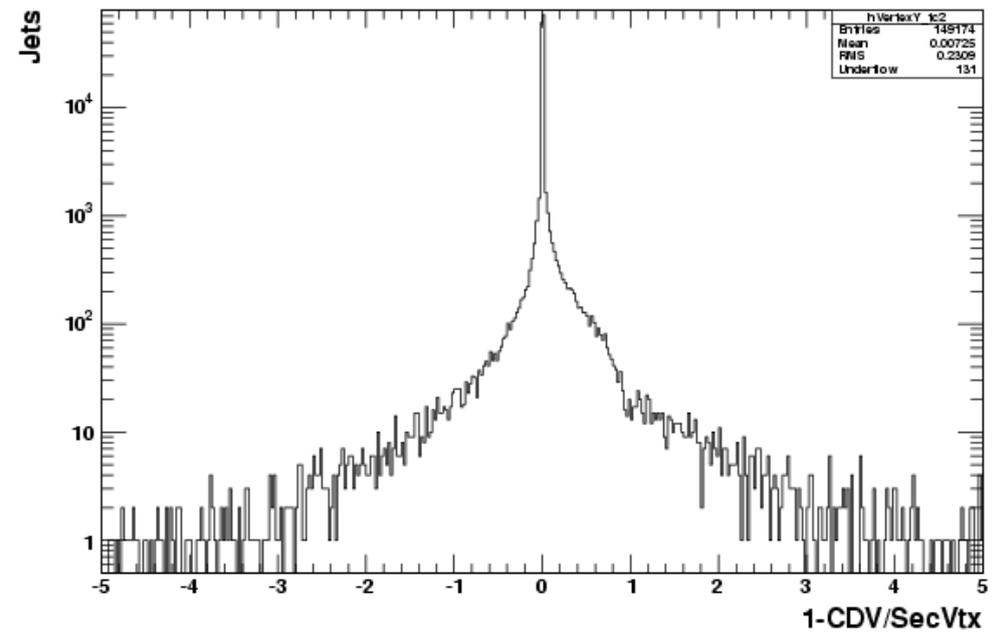
## Vertex X 1-CDV/SecVtx



## Vertex Y SecVtx



## Vertex Y 1-CDV/SecVtx

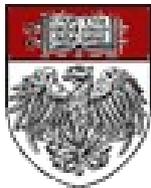


CDV was the working name for the algorithm.

# Conclusions

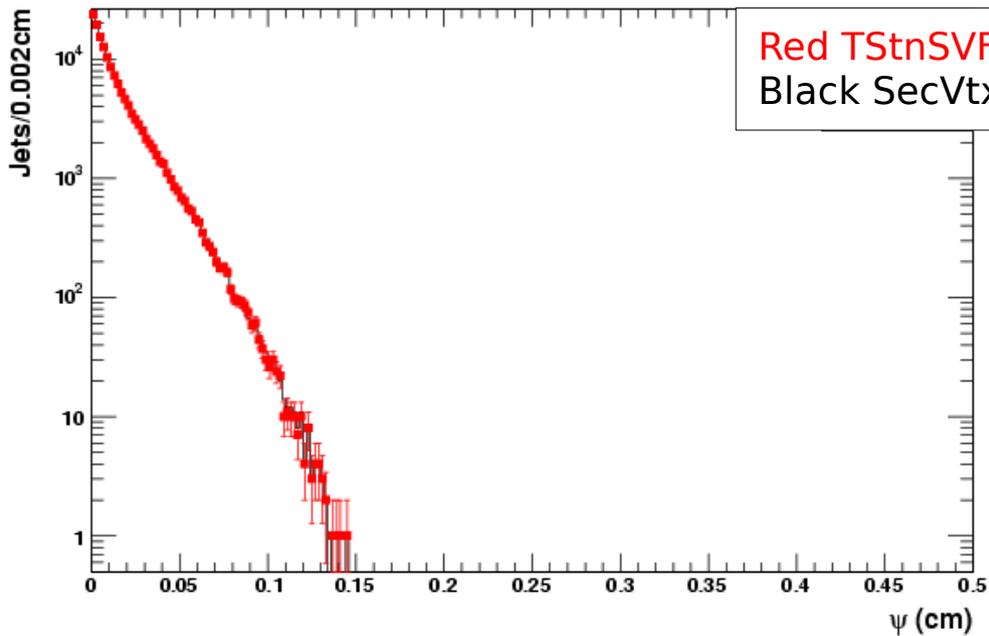
- ▶ My algorithm compares very well to SecVtx, with a high efficiency of matched tags and with only small differences in the essential variables.
- ▶ However, I want to stress that this is not a replacement or a duplication for SecVtx, but is in effect a stand-alone tagger.
- ▶ Next Up: scale factors and mistag matrix

2008-10-08

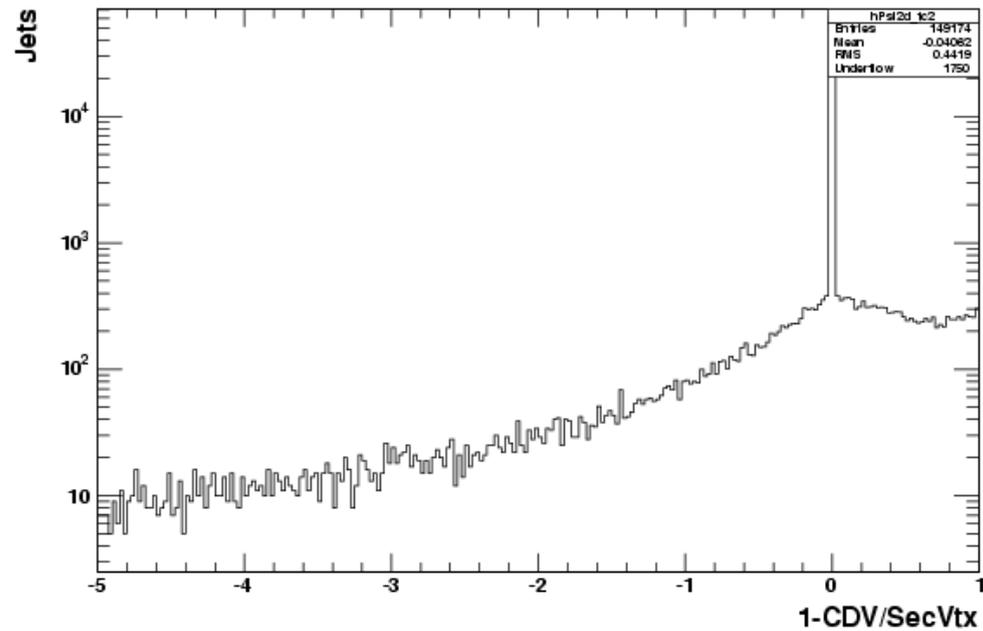


Shawn Kwang

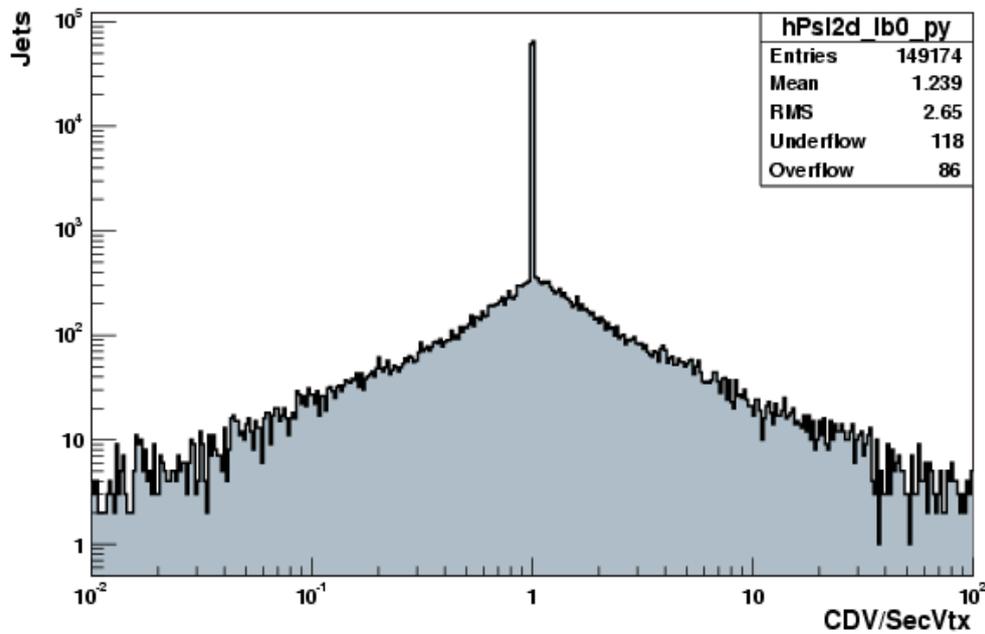
# $\Psi$ 2d SecVtx



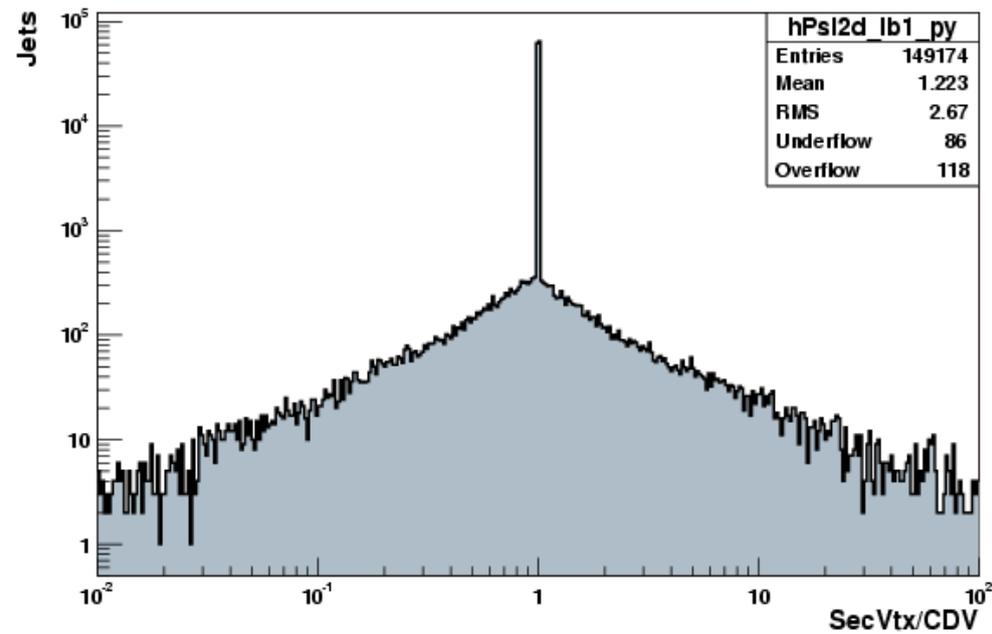
# $\Psi$ 2d 1-CDV/SecVtx



# $\Psi$ 2d



# $\Psi$ 2d



Backup Slide  
CDV was the working name for the algorithm.

